Available online at www.prace-ri.eu

Partnership for Advanced Computing in Europe

Using gsatellite for data intensive procedures between large-scale scientific instruments and HPC

N. Ilieva[a,b,c], Z. Kiss[d]*, B. Pavlov[e,a], G. Szigeti[d]

[a]*National Centre for Supercomputing Applications (NCSA), Sofia, Bulgaria*
[b] *Institute of Information and Communication Technologies, BAS, Sofia, Bulgaria*
[c]*Institute for Interdisciplinary Research and Technology (INIRT), Sofia, Bulgaria*
[d]*KIFU - NIIFP, Budapest, Hungary*
[e]*Sofia University "St. Kl. Ohridski", Sofia, Bulgaria*

Abstract

Being able to handle large volumes of valuable data is a central point in linking large-scale scientific instruments (i.e. satellites, laser facilities, sequencers, accelerators, etc.) with HPC infrastructure. Whether the data is created through experiment or synthetically, it has to be reliably transferred, analysed, stored and archived for further use, or reference, as re-creating it may be expensive, time-consuming or even impossible. This necessitates support improvement for data and computationally intensive applications in terms of data transfer and architecture diversification. We discuss these issues on the example of two large-scale research infrastructures – accelerators (in the context of high-energy physics research and medical applications) and Extreme Light Infrastructure (ELI).

1. INTRODUCTION

The importance of computational methods and processing of Big Data has grown to the extent that they are sometimes called – next to experiment and theory – the pillars of science. On the one hand, constant technological advancement leads to increased precision of scientific instruments used in experimentation. On the other hand, the results of the experiments cannot be analysed in practice without the use of large HPC facilities and advanced software tools. Among the technical challenges that need to be overcome in order for the modern science to fully benefit from the technological advancements, we put forward the following (see also PRACE 4IP Deliverable D6.3 [1]):

- Large-scale instruments produce enormous amounts of data, which needs to be stored and archived for future query and reference;
- Experiments are often augmented with numerical simulations, which themselves require substantial computational power and can generate a comparable amount of data. In this sense, HPC facilities and efficient numerical software together can be viewed as an instrument of their own, and are of fundamental importance in the process of validation and refinement of physical models.

We focus on two particular cases:

- MIC-oriented Multithreading for HEP (high-energy physics) and Health Geant4 Computations
- Supporting ELI (Extreme Light Infrastructure) with HPC

The former is piloting the use of emerging large scale Intel Xeon Phi based HPC facilities for a variety of applications ranging from LHC physics to hadron therapy for cancer treatment. Initially it will focus on

* *Corresponding author: kzoli@niif.hu*

demonstrating the service for LHC experiments at CERN. The basic software in all these fields is Geant4 [2, 3, 4] (GEometry ANd Tracking) – a platform for simulation of the passage of particles through matter [5].

ELI [6] is an ESFRI [7] project to build world leading next generation laser research facilities aiming to analyse interaction between light and matter. This technology can be used to enable new ways of research within material sciences, biochemistry, medicine, nanotechnologies, nuclear physics, astrophysics and cosmology. Particle-in-Cell analysis is one common method to analyze different interaction of light and materials as a research of computational radiation physics. This analysis requires HPC resources.

Most of Geant4 applications, which will be actively used in the next years, are very demanding in terms of computing power and generated data volumes, and the demand will surpass the potential of the available resources. It is important that simulated events are stored and transported reliably. In addition, processing sensitive data (e.g. for hadron therapy treatment of a patient) must meet very strict security policies, which in many cases significantly complicates the workflow.

While detectors within laser facilities are becoming more and more sensitive, the amount of data to be analyzed during a test setup is constantly growing. While the laser facility complex itself has computing resources, in some cases, it will not be able to satisfy the processing requirements of some advanced research use cases. The facilities are planning to utilize external computing infrastructures like PRACE. The large amount of data to be transferred to an external HPC provider and the processed data to be sent back to the facility needs the analysis of new tools capable of scheduling the data transfer in both directions while interacting with the HPC scheduler to make offloading of processing fully automatic.

Based on the requirements described for both use cases, the workflows would benefit from a reliable data transfer job scheduler, like gsatellite to enable fast transfer of multiple datasets with fault tolerance and a notification system. Combined with other wrapper tools, like gtransfer, the setup would allow transfer tasks to be more robust than using common transfer tools to handle transfer of each files separately.

2. SOLUTION DESIGN

2.1. Gsatellite introduction

Gsattelite [8] is a promising candidate for a job scheduler for services on hybrid architectures including Intel Xeon Phi co-processors. Gsatellite provides a variety of functionalities and is user friendly. It is portable and based on a few Bash scripts, so it can be modified if necessary and there are no problems with the support, which ensures the long-term sustainability of the implemented services. Gsatellite provides also some useful features, which are not supported by all job schedulers – for example, putting jobs temporary on hold (using gsatellite commands gqhold, and gqrls).

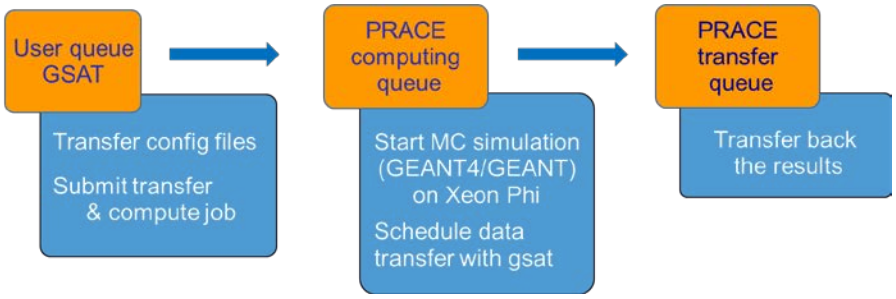

Figure 1. *Usage of gsatellite in the the service workflow.*

Gsatellite proved useful for performing data transfer scheduling, using gtransfer or xrdcp utility of XRootD [9], but it can also be used with any other convenient file-transfer utility.

For heavily loaded systems it might be important to set job priorities and/or time limits for job execution. Profiling the resource usage per user is also important for fair access to the recourses. Both options are not implemented in gsatellite.

The responsibilities handled by gsatellite in the implemented services are (see Fig. 1):

- Transfer input data to remote HPC facility
- Schedule compute job on HPC machine

- Monitor remote job
- Transfer output file back home

The reliability of the service requires properly handling failed jobs. If the failure is due to wrong user input, nothing can be done by the service but to send an error message to the user. However if the failure is due to some malfunctions of the service the reliability requires that the job is automatically rerun (see Fig. 2):

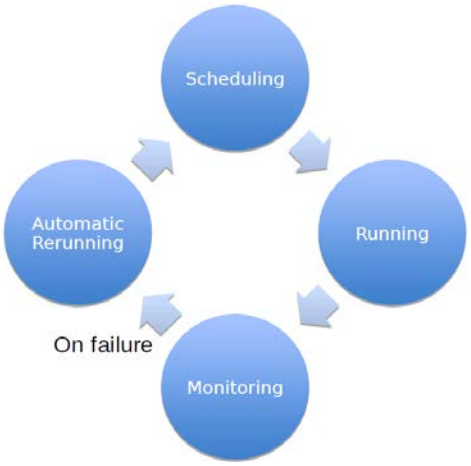

Figure 2. *Handling a job failure by the gsatellite scheduler.*

3. TECHNICAL DISCUSSION: USE CASES

3.1 Gsatellite in GEANT4@MIC service

The GEANT4 service on Xeon Phi is described in detail in [10]. Here we focus on the use of gsatellite as an auxiliary tool used to provide the high quality and reliable GEANT4 service.

GEANT4 is a Monte Carlo package to simulate the passage of ionizing radiation and particles trough the matter. Thus, the input data as a rule is only a configuration file while the output could be a huge file with simulated physical events Figure 3.

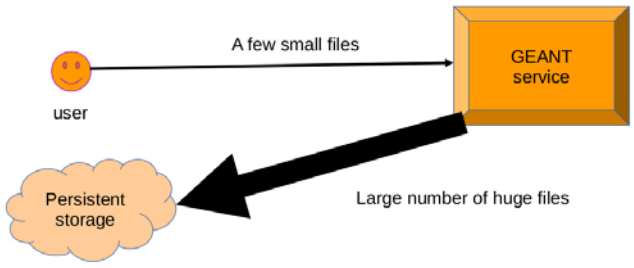

Figure 3. *Asymmetric bidirectional transfer: from users to GEANT service – only a few small size configuration files, while from service to users – huge data files with MC data.*

The service design is inspired by Cloud technologies, i.e. sending the request and getting automatically the job completed. In our particular case, the request is send by uploading the configuration files to the service storage that triggers all the procedures and the user receives back the result (output of the GEANT4) automatically (Figure 4). This is done by means of a custom program, which checks the input directory for new files. Two files are expected to be present – a file with a configuration for GEANT4 job and one auxiliary configuration file needed by GEANT4 service. The GEANT4 configuration file uses a special syntax described in detail in the GEANT4 manual. The auxiliary file is designed to provide some information needed by the service in order to function properly.

The auxiliary file contains the following information, relevant for data transfer:

- The file transfer protocol. If none is specified, no automatic transfer is performed and user could transfer the data manually later on
- The path to the storage where the result should automatically be transferred
- E-mail address for automatic notification

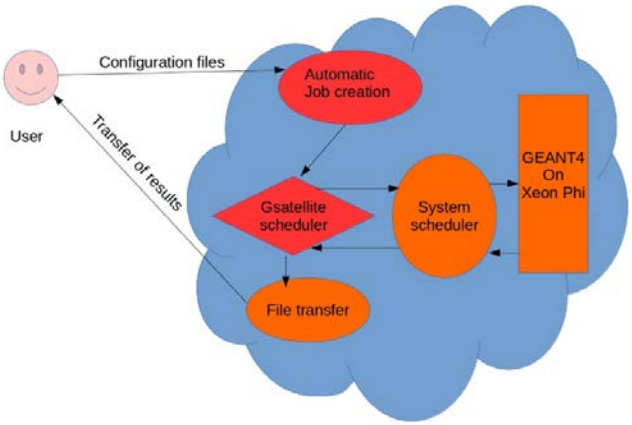

Figure 4. *Gsatellite as used for the designated GEANT4@MIC service*

When the two configuration files are detected the automatic procedure starts (Fig. 4). A job is automatically created and scheduled by gsatellite. A Python script running in a background on the front-end node carries out both detection and job creation. Gsatellite in turn sends the computation job to the system scheduler. The system scheduler is preinstalled on the supercomputer and is responsible to share the resources between services and users. Currently, the GEANT4 service is not supposed to bypass the system scheduler on the test system Avitohol [11] and to access the computation nods directly – the access is through the system scheduler. After the end of computing job, control is back to gsatellite job, which schedules and executes data transfer. Detailed workflow performed by the gsatellite is shown in Fig. 5. Gsatellite starts the simulation, monitors its progress and handles data transfer. In case of service failure the job is automatically resubmitted.

At present, the only supported protocols for automatic transfer are SFTP and gtransfer [12]. With SFTP, the credentials (keys) can be permanently set up upon user account creation, enabling an automatic transfer latter on, while the grid proxy used for gtransfer needs to be recreated, because it expires after a given time. In certain cases, however, this might be considered as an advantage and not disadvantage for gtransfer protocol. The user has to re-create the proxy, before the file transfer by invoking grid-proxy-init command form Globus Toolkit [13], which will prompt for a password. Should the transfer fail for some reason (expired grid proxy, etc.), the files are kept until the user transfers them safely. Successfully transferred files are cleaned.

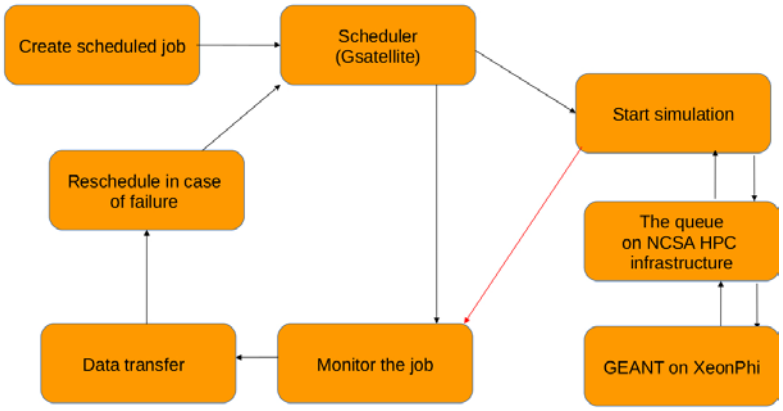

Figure 5. *Detailed workflow with Gsatellite as a service scheduler.*

The recommended transfer is via gtransfer, which is supported by PRACE. SCP is added to extend the service to potential users lacking GRID experience and infrastructure. In the pilot implementation, the Globus Toolkit, gtransfer, UberFTP and tgftp [14] are installed on the service front-end and are accessible for the users.

3.2 Using gsatellite between ELI and PRACE

The workflow supporting transfers between ELI sites and PRACE sites uses core software described in PRACE service catalogue apart from gsatellite, which is a service that has passed through the PRACE security review process. The solution initiates transfer on PRACE HPCs, and needs only standard gridFTP on the ELI side. It uses Globus transfer solution, which supports gsiftp:// (GridFTP), ftp://, http://, https://, and file:/// protocols (see Fig. 6).

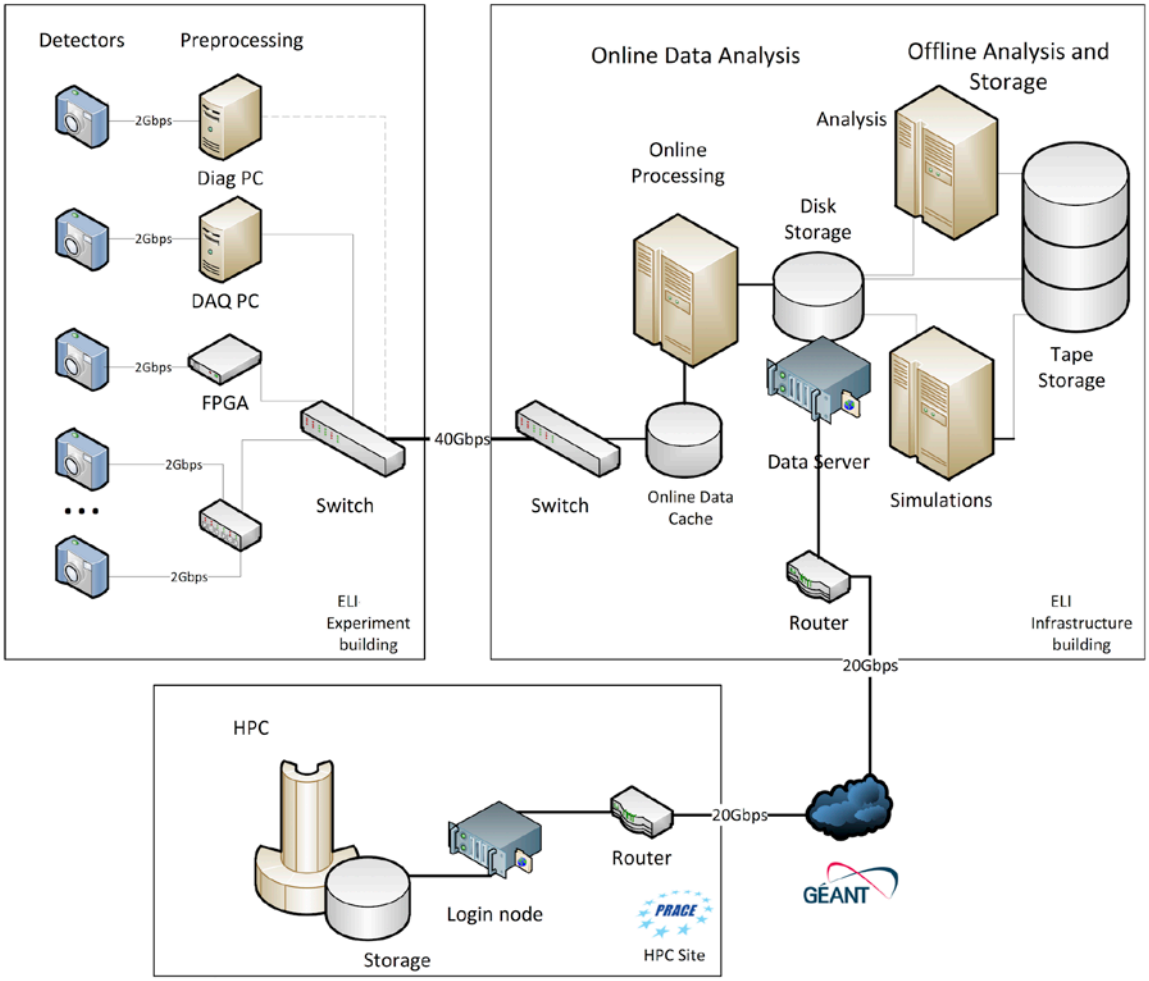

Figure 6. *Test setup including ELI and PRACE local infrastructure*

This workflow is similar to the GEANT4 service but it tries to solve the problem using only PRACE-supported solutions, see Fig. 7. The remote input file, e.g. a gridftp location needs to be specified in the workflow script, and this script then is submitted to gsatellite using gsatctl command.

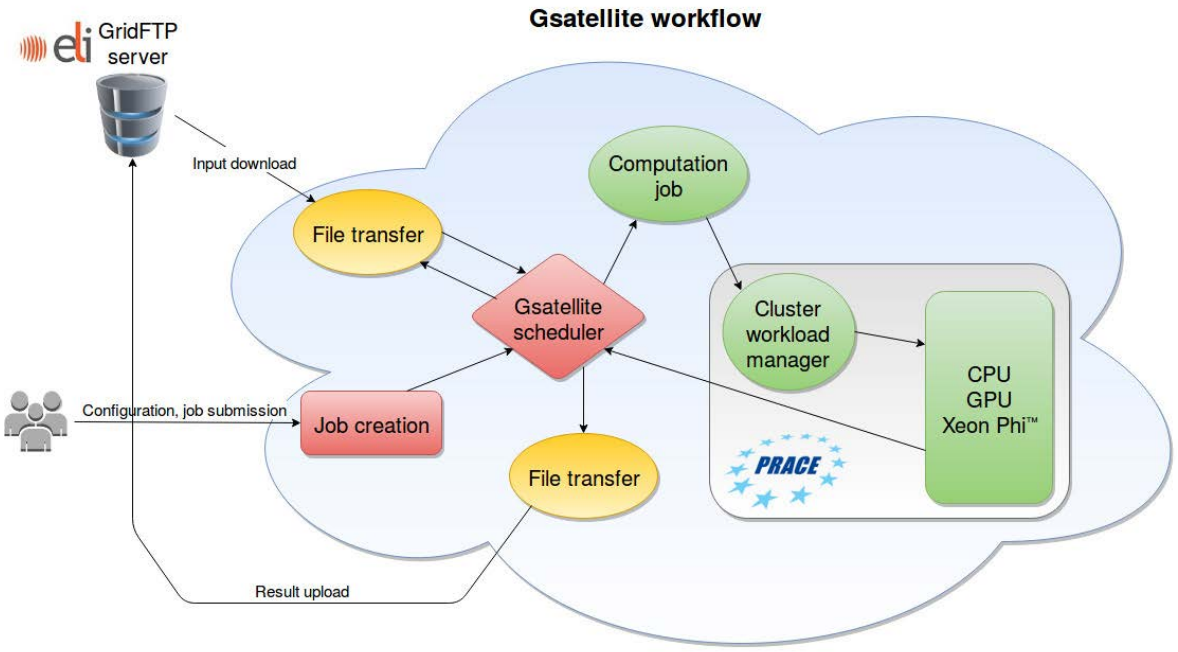

Figure 7. *Using gsatellite between ELI and PRACE*

The workflow will execute the following steps (see Fig. 8):

1. Transfer input data from the ELI gridFTP server to target HPC. Before the transfer it creates a proxy certificate using the credentials of the user initiating the transfer. This account needs to have write access on the remote server. After successful transfer, it submits a computation job to gsatellite.
2. This step submits a computation job to the SLURM workload scheduler (it can be adapted to PBS, Torque, etc. because the job script is uniform). Afterwards, local scheduler executes a submitted computation on the requested resource. After successful execution, the final command of the computation job submits a new gsatellite transfer job that will transfer back the result to the previously configured remote location.
3. In the last step, gsatellite transfers results to the ELI gridFTP server. It creates a proxy certificate using the credentials of the account executing the script to be used throughout the transfer of output data.

   Each internal gsatellite jobscript is generated and submitted automatically by the previous step otherwise the workflow will stop if it receives an error signal. Notification of gsatellite or HPC scheduler system notifies users on failure.
   One can schedule not only a transfer job with gsatellite, but also a compute job. The advantage of this feature is that the offloading batch jobs (transfer input, compute, transfer output back) could be executed in parallel.
   A proxy is generated before each transfer to ensure valid credentials at all times.

3.3 Running Gtransfer via gsatellite

i) Transfer job creation

For creating a job, the actual gtransfer command has to be stored in a dedicated file (gt.job in the example) with the following content:

```
#!/bin/bash
#GSAT -T gtransfer
gt -s host_url/path_to_files/* -d  destination_url/path_to_storage/  -e –v
```

The flags -e -v are optional. The files to be transferred are listed in URL format after -s flag, while the destination is listed in URL syntax after -d flags. The source URL is usually a file, while the destination URL is usually a storage server (for example gsiftp:///server_url/storage_dir).

ii) Starting the job scheduler

```
gsatlcd –start
```

Before starting the gtransfer jobs, some additional environment variables (GLOBUS_LOCATION should point to actual Globus Toolkit installation – e.g. /usr/local/globus-5.0.2) have to be specified:

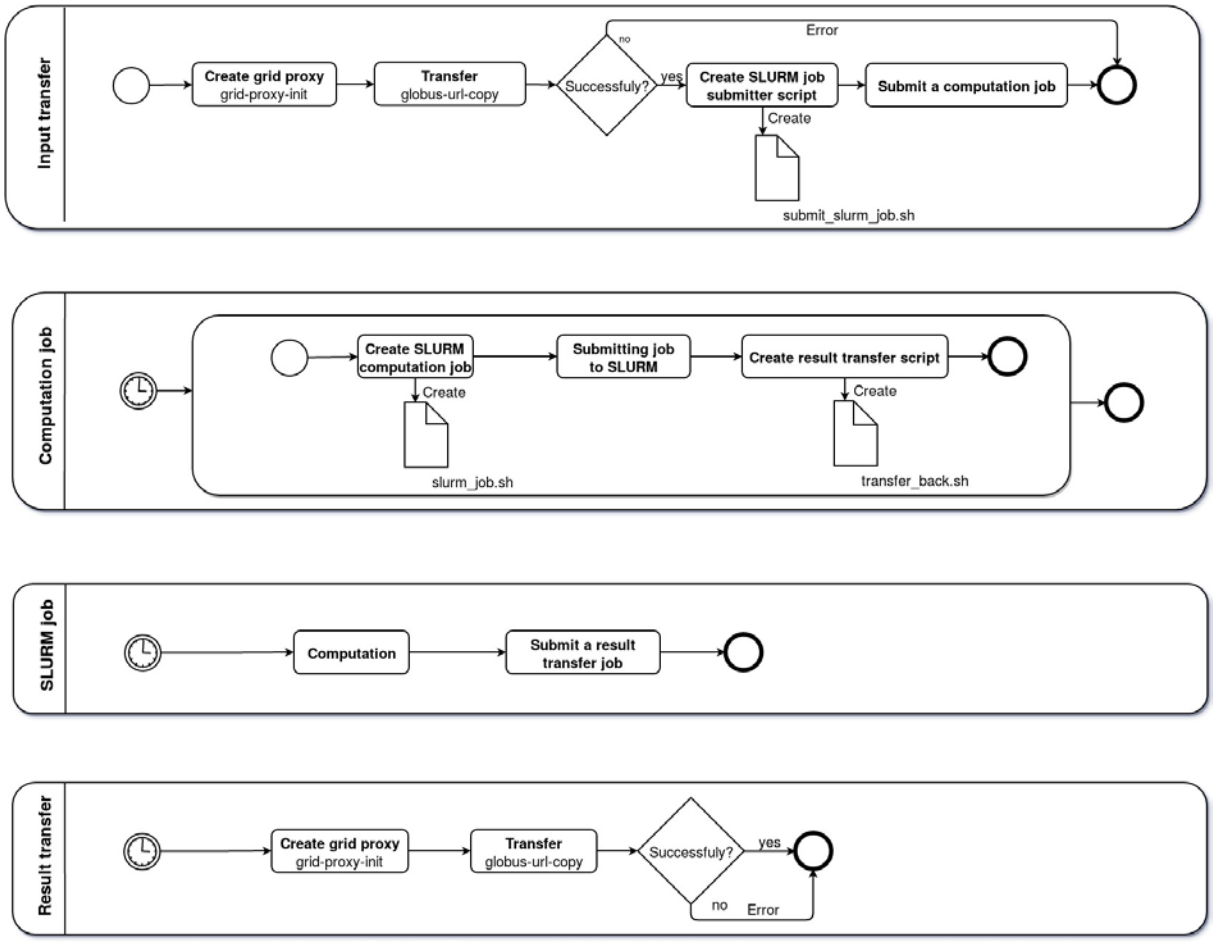

Figure 8. *Flowchart of ELI workflow*

```
export GLOBUS_LOCATION=/usr/local/globus-5.0.2
export LD_LIBRARY_PATH=$GLOBUS_LOCATION/lib
.$GLOBUS_LOCATION/etc/globus-user-env.sh
export PATH=$HOME/opt/uberftp/bin/:${PATH
```

iii) Submitting transfer job

`gqsub gt.job` or alternatively `gsatctl --qsub gt.job`

The gqsub command displays the id number of the job, which can be used to follow the job execution.

*iv) **Checking job status and following the job execution (for running jobs)***

```
gqstat    or alternatively      gsatctl –qstat

tail -f $HOME/.gsatellite/gschedule/jobs/00025.d/job.stdout
```

The log files for each job are kept in subfolders in $HOME/.gsatellite/gschedule/jobs/ folder. In case of job failure, some hints for the possible reasons can be found in the job.stderr file.

The scripts (i) – (iii) are generated and launched automatically by the service when a new configuration files are uploaded by the user on the service input area. A dedicated python script checks for newly uploaded files and if it founds any the automatic procedure is triggered – job files are parsed and are submitted to gsatellite scheduler.

CONCLUSIONS

To summarise, the usage of gsatellite in linking HPC facilities to large-scale scientific instruments is justified, the main strengths and weaknesses being listed in Table 1.

Table 1. *Experiences with gsatellite*

Strengths	**Weaknesses**
Gsatellite was evaluated by PRACE security team and found to be secure	There are universal data transfer tools available designed to do similar staging, and transfer of data (e.g. B2STAGE [15]), but requires additional software to be installed
Written in Bash – easy to modify, port, maintain	Transfer is user based (needs user keys or certificates).
Gtransfer is already an additional PRACE service	Transfer and compute are not integrated, need further integration
Able to use different transfer tools apart from Gtransfer. The variety of potential tools is beneficial for users lacking GRID infrastructure and certificates	Needs to “communicate” with preinstalled system scheduler in order to execute computation jobs
Provides scheduled and automatic data transfer to/from an HPC facility	Actual offloading to external resource is not yet automated
Actual workflow is achieved by submission of transfer and compute jobs	
Provides job monitoring, retries transfer and notifies user on error	
Uses standard X509 certificate based authentication, which is standard at PRACE and compatible with EGI, EUDAT, ELI infrastructures	
Supports encryption when used with Gtransfer	

In both use cases gsatellite proved to be a lightweight data transfer scheduler solution already validated by PRACE, and using it in a framework requires no additional tools to be installed near the partner infrastructure. The framework using gsatellite provides reliable transfer of multiple large files between the scientific equipment and compute site, along with automated computations for specific use cases without user interaction. The framework is especially valuable when an input parameter or data changes, while the code itself does not.

Workarounds have been successfully implemented to cope with its weaknesses (automatic submission of jobs, etc.).

The framework requires evaluation in practice by potential users. The test system Avitohol will be sufficient for sustaining the GEANT4 pilot service, while NIIF will continue to test the framework with ELI on LEO, the PRACE Tier-1 machine.

References

[1] PRACE-4IP D6.3 deliverable http://www.prace-ri.eu/IMG/pdf/D6.3_4ip.pdf
[2] S. Agostinelli et al., NIM, vol. **A 506** (2003) 250-303; http://www.sciencedirect.com
[3] J. Allison et al., IEEE Transactions on Nuclear Science **53** (2006) 270-278; http://ieeexplore.ieee.org
[4] J. Allison et al., NIM, vol. **A 835** (2016) 186-225; http://www.sciencedirect.com
[5] https://geant4.web.cern.ch/geant4/https://geant4.web.cern.ch/geant4/
[6] Extreme Light Infrastructure (ELI): https://eli-laser.eu/
[7] ESFRI: http://ec.europa.eu/research/infrastructures/index_en.cfm?pg=esfri
[8] https://github.com/fr4nk5ch31n3r/Gsatellite/wiki/Gsatellite-explained
[9] http://xrootd.org/doc/man/xrdcp.1.html
[10] N. Ilieva, B. Pavlov, P. Petkov, and L. Litov, GEANT4 on Large Intel Xeon Phi Clusters: Service Implementation and Performance, PRACE Whitepaper (number and link to be added)
[11] http://www.hpc.acad.bg/system-1/
[12] https://github.com/fr4nk5ch31n3r/gtransfer/
[13] Globus Toolkit: http://toolkit.globus.org/
[14] The GridFTP benchmark, test and transfer script https://github.com/fr4nk5ch31n3r/tgftp
[15] EUDAT B2STAGE service description: https://eudat.eu/services/userdoc/b2stage

Acknowledgements

This work was financially supported by the PRACE project funded in part by the EU's Horizon 2020 research and innovation programme (2014-2020) under grant agreement 653838.

APPENDIX

Gsatellite installation

Gsatellite does not require additional packages to be installed in advance, but only a bash shell. Here are gsatellite installation instructions:

```
mkdir -p tmp && cd tmp && wget -c
https://github.com/fr4nk5ch31n3r/Gsatellite/archive/master.tar.gz
tar -xzf master.tar.gz
cd Gsatellite-master && ./install.sh
```

Gsatellite should be started again after each login.

```
gsatlcd --start
```

It is handy to add this line to the login script (for instance in bashrc file).

Gtransfer installation

Gtransfer requires the installation of several additional packages:

- gtransfer
- tgftp
- Globus Toolkit
- UberFTP

Installation of tgftp:

```
wget -c https://github.com/fr4nk5ch31n3r/tgftp/archive/master.tar.gz
mv master.tar.gz tgftp.tgz
tar -xzvf tgftp.tgz
cd tgftp-master/
./install.sh
```

Installation of Globus Toolkit:

```
wget http://toolkit.globus.org/ftppub/gt5/5.0/5.0.2/installers/src/gt5.0.2-all-
source-installer.tar.gz
tar -xzvf gt5.0.2-all-source-installer.tar.gz
mkdir $HOME/opt/globus-5.0.2/
export GLOBUS_LOCATION=$HOME/opt/globus-5.0.2
 cd gt5.0.2-all-source-installer
./configure –prefix=$GLOBUS_LOCATION
make
make install

export LD_LIBRARY_PATH=$GLOBUS_LOCATION/lib
. $GLOBUS_LOCATION/etc/globus-user-env.sh
```

Installation of UberFTP (after Globus Toolkit has been installed):

```
wget http://pkgs.fedoraproject.org/repo/pkgs/uberftp/uberftp-client-
2.5.tar.gz/0415a8b2584ec5c47b879f503ee002b9/uberftp-client-2.5.tar.gz
tar -xzvf uberftp-client-2.5.tar.gz
cd uberftp-client-2.5
mkdir $HOME/opt/uberftp
./configure --with-globus=$GLOBUS_LOCATION --with-globus-flavor=gcc64dbg –
prefix=$HOME/opt/uberftp
make
make install
export PATH=$HOME/opt/uberftp/bin/:${PATH}
```

Gtransfer is installed with the following commands:

```
wget -c https://github.com/fr4nk5ch31n3r/gtransfer/archive/master.tar.gz
mv master.tar.gz gtransfer.tgz
tar -xzvf gtransfer.tgz
cd gtransfer-master/
./install.bash
```